

Appendix ASource Code

```
1
2
3  /**
4   * Superclass for all classes
5   */
6  import java.io.*;
7  class Super {
8      static FileOutputStream fos;
9      public static void trace(String s) {
10         System.out.println(s);
11     }
12 }
13
14 /**
15  * Superclass for all record types. Provides methods to read adata in buffer.
16  */
17 import java.io.*;
18 import java.lang.reflect.*;
19 import java.lang.Class.*;
20
21 class Record extends Super{
22     int[] buffer;
23     int length;
24
25     /**
26      * reads data from the sysadata file and fills the objects buffer
27      */
28     public void fillBuffer(FileInputStream fis, int _length) {
29         buffer = new int[_length];
30
31         for (int i= 0; i<_length;i++){
32             try{
33                 buffer[i] = fis.read();
34             }catch(IOException e){
35                 trace("Error in "+this.getClass()+" "+ e);
36             }
37         }
38     }
39 }
40
41 /**
42  * SysAdata reads and interprets a COBOL Associated Data (ADATA) file
43  * and creates the XMI document.
44  */
45 import java.io.*;
46 import java.util.Vector;
47 import com.ibm.ims.message.*;
48
49 public class SysAdata extends Super{
50     static FileInputStream fis = null;
```

```

1  static CHSRecord chs_rec;
   static CURecord cu_rec;

2  static Vector symbolVector = new Vector();
   static Id model = null;
3  static int rootLength = 0;
   /**
   * addID adds one or more DataItem object to the model recursevily
   */
4  public static int addID(Id parent, int symbolNumber){
       SymbolRecord symbol = (SymbolRecord) symbolVector.elementAt(symbolNumber);
5       try{
           trace("adding "+symbol.getName());
           Id id = parent.add(Type.DATA_ITEM, null);
6           id.set(Property.NAME, symbol.getName());
           id.set(Property.TYPE, symbol.getAttribute());
7           id.set(Property.LENGTH, symbol.getLength());
           id.set(Property.HAS_VALUE, ""+symbol.hasValue());
8           if(symbol.hasValue()){
               rootLength = rootLength + symbol.getSize();
               id.set(Property.VALUE, "");
9           }

           symbolNumber++;
10          if((symbol.hasValue() == false) && (symbolNumber < symbolVector.size())){
               SymbolRecord nextSymbol = (SymbolRecord)
11          symbolVector.elementAt(symbolNumber);
               if(nextSymbol.redefines()) {
                   symbolNumber = addID(id,symbolNumber);
12               }
               else {
13                   while ((symbol.getLevel() < nextSymbol.getLevel()) && (symbolNumber
< symbolVector.size())){
14                       symbolNumber = addID(id,symbolNumber);
                       if(symbolNumber < symbolVector.size())
                           nextSymbol = (SymbolRecord)
15          symbolVector.elementAt(symbolNumber);
                       }
16                   }
               }catch(Exception e){e.printStackTrace();}

17          // returns the number of the next symbol to be processed
           return symbolNumber;
18      }

   public static void main(String args[]){
       boolean loopflag = true;
19       int counter = 0;

       if(args.length > 1){
20           //open adata file which is given as first argument in the parameter list
           try {
21               fis = new FileInputStream(args[0]);
               //fis = new FileInputStream("d:/jan/parser/sysadata/data.adt");

```

```

1      }
2      catch(java.io.FileNotFoundException e){
3          System.out.println("error" + e);
4          return;
5      }
6
7      //reads first common header section (CHSRecord) from file
8      chs_rec = new CHSRecord(fis);
9
10     /**
11     * loop reads records from the adata file
12     * checks for CURecord, SymbolRecord and skips all other records
13     * stores SymbolRecords in symbolVector
14     */
15     do{
16         switch (chs_rec.getrecordType()){
17             case 2: {
18                 //Compilation Unit Record, 02hex
19                 cu_rec = new CURecord(fis);
20                 if(cu_rec.getType() == 1)
21                     loopflag = false;
22                 chs_rec.fill(fis);
23                 break;}
24             case 66: {
25                 //Symbol Record, 042hex
26                 SymbolRecord symbolrecord = new
27                 SymbolRecord(fis,chs_rec.getrecordLength());
28                 if(symbolrecord.getType()==64)
29                     symbolVector.addElement(symbolrecord);
30                 chs_rec.fill(fis);
31                 break;}
32             default: {
33                 //all other records are skipped
34                 skip(chs_rec.getrecordLength());
35                 chs_rec.fill(fis);
36                 break;}
37         }
38     }while(loopflag);
39
40     try{
41         fis.close();
42     }catch(IOException e){trace(""+e);}
43
44     try {
45         model = Model.instance().getSession();
46         Id root = model.add(Type.DATA_ITEM, null);
47         root.set(Property.NAME, "ROOT");
48         root.set(Property.TYPE, "ROOT");
49         root.set(Property.HAS_VALUE, "False");
50
51         do{
52             counter = addID(root, counter);
53         }while(counter<symbolVector.size());

```

```

1          root.set(Property.LENGTH, ""+rootLength);
          Model.instance().save(root, args[1], Model.DEFAULT, new java.util.Vector());
          //Model.instance().save(root, "sample.xml", Model.DEFAULT, new java.util.Vector());
2      } catch (Exception e) {e.printStackTrace();}
    }
3  else
        trace("No input and output file given\nSyntax is: java SysAdata input.adt output.xml");
    }
4  /**
    * skips the given amount of bytes in the SysAdata input file.
    */
5  public static void skip(int length) {
        try{
6          fis.skip(length);
          }catch(java.io.IOException e){trace(""+e);}
7  }
8  /**
    * CHSRecord: Common header Section - x0001
    * 12byte long, common for all record types
    */
10 import java.io.*;
    class CHSRecord extends Record{
11     public byte langCode; // Language Code
    public short recordType; // Record Type
    public byte sysadataLevel;// SysAdata Architecture Level
12     public byte contFlag; // bit 1: record is continued; bit 2: integers are Little-Endian; bits 3-8: reserved
    public byte editionLevel; // Indicates a new format for a specific record type; usually 0
13     public int reserve; // Reserved for future use
    public short recordLength; // Record Length following header (in bytes)
14     /**
    * Constructor
    */
15     public CHSRecord(FileInputStream fis){
16         fill(fis);
17     }
    /**
    * fills all fields of the Common Header Section Record
    */
18     public void fill(FileInputStream fis) {
19         super.fillBuffer(fis,12);
        langCode = new Integer(buffer[0]).byteValue();
        recordType = new Integer(buffer[1]+(buffer[2]*256)).shortValue(); //xchanges byte 2 and 3; shifts
20         byte3 left
        sysadataLevel = new Integer(buffer[3]).byteValue();
        contFlag = new Integer(buffer[4]).byteValue();
        editionLevel = new Integer(buffer[5]).byteValue();
21

```

```

1      recordLength = new Integer(buffer[10]+(buffer[11]*256)).shortValue();
2      }
3      /**
4       * returns length of the following record
5       */
6      public short getrecordLength(){
7          return(recordLength);
8      }
9      /**
10     * returns type of the following record in decimal
11     */
12     public short getrecordType(){
13         return(recordType);
14     }
15     /**
16     * prints out attributes of CHSRecord
17     */
18     public void print(){
19         trace("langCode " + langCode + ";recordType " +recordType+ " ;sysadataLevel "+sysadataLevel+
20         ";contFlag "+contFlag+ ";editionLevel "+editionLevel+ ";recordLength "+recordLength);
21     }
22     }
23     /**
24     * Compilation Unit Start/End Record - x0002
25     * 8 bytes long
26     */
27     import java.io.*;
28
29     class CUREcord extends Record{
30         short type;
31     }
32     /**
33     * CUREcord constructor
34     */
35     public CUREcord(FileInputStream fis) {
36         fill(fis);
37     }
38     /**
39     * fills all needed fields of the Compilation Unit Start/End Record
40     */
41     public void fill(FileInputStream fis) {
42         super.fillBuffer(fis,8);
43
44         type = new Integer(buffer[0]+(buffer[1]*256)).shortValue();
45     }
46     /**
47     * returns the type of the CUREcord
48     * compilation unit types are: x0000=Start, x0001=end
49     */
50     public int getType() {
51         return type;
52     }
53     }

```

```

1  /**
2   * Symb IRecord: Common header Section - x0001
3   * variable length, contains description of all symbols
4   *
5   * All get methods are used to fill the DataItems in the XMI Document.
6   */
7
8  import java.math.*;
9  class SymbolRecord extends Record {
10     //fields of the Common Header Section
11     int symbolId;
12     byte level;
13     byte symbolType;
14     byte symbolAttribute;
15     byte[] clauses = new byte[1];
16     BigInteger clausesB;
17     byte[] flags1 = new byte[1];
18     BigInteger flags1B;
19     int size;
20     int parentId;
21     int redefinedId;
22     short symbolNameLen;
23     String symbolName;
24
25     //symbolAttributeValues are put into the attribute tag of the DataItem
26     static String symbolAttributeValues[] = {
27         "",
28         "Numeric",
29         "Alphanumeric",
30         "Group",
31         "Pointer",
32         "IndexDataItem",
33         "IndexName",
34         "Condition",
35         "", "", "", "", "", "", "", "",
36         "File",
37         "SortFile",
38         "", "", "", "", "", "", "", "",
39         "ClassName",
40         "ObjectReference"};
41
42     /**
43     * Constructor
44     */
45     public SymbolRecord(java.io.FileInputStream fis, int length) {
46         fill(fis, length);
47     }
48     /**
49     * fills needed fields of Symbol Record
50     */
51     public void fill(java.io.FileInputStream fis, int length) {
52         super.fillBuffer(fis,length);
53     }
54 }

```

```

1      symbolId = new Integer(buffer[0] + buffer[1]*16*16 + buffer[2]*16*16*16*16 +
buffer[3]*16*16*16*16*16*16).intValue();
2      level = new Integer(buffer[8]).byteValue();
3      symbolType = new Integer(buffer[10]).byteValue();
4      symbolAttribute = new Integer(buffer[11]).byteValue();
5      clauses[0] = new Integer(buffer[12]).byteValue();
6      clausesB = new BigInteger(clauses);
7      flags1[0] = new Integer(buffer[13]).byteValue();
8      flags1B = new BigInteger(flags1);
9      size = new Integer(buffer[20] + buffer[21]*16*16 + buffer[22]*16*16*16*16 +
buffer[23]*16*16*16*16*16*16).intValue();
10     parentId = new Integer(buffer[44] + buffer[45]*16*16 + buffer[46]*16*16*16*16 +
buffer[47]*16*16*16*16*16*16).intValue();
11     redefinedId = new Integer(buffer[48] + buffer[49]*16*16 + buffer[50]*16*16*16*16 +
buffer[51]*16*16*16*16*16*16).intValue();
12     symbolNameLen = new Integer(buffer[90] + buffer[91]*16*16).shortValue();

13     char[] temp = new char[symbolNameLen];
14     int j=0;
15     for(int i=104;i<(104+symbolNameLen);i++){
16         temp[j]=(char)buffer[i];
17         j++;
18     }
19     symbolName = new String(temp);
20 }
21 public String getAttribute(){
    return symbolAttributeValues[symbolAttribute];
}
22 public String getLength(){
    return new Integer(size).toString() ;
}
23 public int getLevel(){
    return level;
}
24 public String getName(){
    return symbolName;
}
25 public int getParentId(){
    return parentId;
}
26 public int getSize(){
    return size;
}
27 public int getSymbolId(){
    return symbolId;
}
28 public int getType(){
    return symbolType;
}
29 }
30 /**
31  * bit 7 is 1 if the symbol is redefined
32  * symbol is group if symbolAttribute == 3
33  */

```

```
1 public boolean hasValue(){
2     if(flags1B.testBit(7) || symbolAttribute == 3)
3         return false;
4     else
5         return true;
6 }
7 public void print() {
8     trace("symbolID:"+ symbolId+ " , level:"+ level+", parentId:"+parentId+", size:"+size+",
9     redefined:"+flags1B.testBit(7)+", symbolAttribute:"+symbolAttribute+", redefinedId:"+redefinedId+",
10    name:"+symbolName+" ,hasValue:"+hasValue());
11 }
12 /**
13  * bit 5 is high if this symbol redefines another one
14  */
15 public boolean redefines(){
16     return clausesB.testBit(5);
17 }
18 }
19
20
21
```


Appendix B

Classes

In various embodiments, each record type is implemented as a class that understands how to handle the binary data from the SysAdata file 78 and provide the data to other classes via defined access methods. Figure 9 illustrates the relationship among the classes. A more detailed description on attributes and methods of each class follows. The source code of all classes may be found in Appendix A.

Class Super

This class offers methods and attributes that are needed by all classes.

Attributes

No Attributes defined.

Methods

- public static void trace(String s)
trace is used to trace error statements. It replaces Java's System.out.print().

Class Record

This class offers methods and attributes that are needed by all Record classes.

Attributes

- int[] buffer
Array of ints that stores the actual data from the SysAdata file 78.
- int length
Stores the size of the buffer array.

Methods

- public void fillBuffer(FileInputStream fis, int _length)

Fills the buffer array.

Class SysAdata

SysAdata wraps methods and attributes to read the SysAdata file 78 and create the XML document template 58. Initially, the complete SysAdata file 78 is processed and all data-entry symbols are saved in the symbolVector. Thereafter, the symbolVector is processed and an XMI model that resembles the document is created.

Attributes

- FileInputStream fis

Wraps the input adata file.

- CHSRecord chs_rec

Is used to temporarily save a CHSRecord.

- CURecord cu_rec

Is used to temporarily save a CURecord.

- Vector symbolVector

The symbolVector stores all SymbolRecords that are extracted from the adata file.

- Id model

model is the root element for the XMI toolkit object hierarchy.

- int rootLength

Stores the added up length of all DataItem.VALUES.

Methods

- public static void main(String args[])

The main method contains the actual loop, in which the SysAdata file is processed. The assembly of the model for the XML document template 58 is initiated here as well.

- public static int addID(Id parent, int symbolNumber)

Recursive method to assemble the model that is then saved as XML.

Parameters are:

- the parents Id, such that the next symbol can be added on the right level.
- the position number of the symbol to be added in the symbolVector.

addID returns the index of the next SymbolRecord object to be processed.

- public static void skip(int length)

Replaces FileInputStream.skip().

Class CHSRecord

Throughout the processing of the SysAdata file 78, each common header section instantiates this class.

Attributes

A description of these attributes is found in Table 1.

- byte langCode
- short recordType
- byte sysadataLevel
- byte contFlag
- byte editionLevel
- public int reserve
- public short recordLength

Methods

- public CHSRecord(FileInputStream fis)

The constructor instantiates the CHSRecord class. The FileInputStream object is handed over from the calling instance.

- public void fill(FileInputStream fis)

Calls Record.fillBuffer() and then extracts the binary data stored in buffer[] into the referring attribute.

- public short getrecordLength()

Returns the length of the following record. Is needed in one embodiment to process the input file correctly.

- public short getrecordType()

Returns the record type.

Class CURecord

A compilation unit start/end record instantiates this class. The CURecord class is used to control the processing of the SysAdata file 78. In certain embodiments, if getType() returns 1, the processing is stopped.

Attributes

- short type

Methods

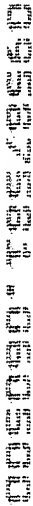
- public CURecord(FileInputStream fis)

The constructor instantiates the CURecord class. The FileInputStream object is handed over from the calling instance.

- public void fill(FileInputStream fis)

Calls Record.fillBuffer() and then extracts the binary data stored in buffer[] into the referring attribute.

- public int getType()



Class SymbolRecord

Any symbol record found during the processing of the SysAdata file 78 instantiates this class.

Attributes

A detailed description of these attribute is given in Table 3. The clauses and the flags1 attribute are converted into BigIntegers and are therefore stored in an array with size one to provide the correct input to the BigInteger constructor. The BigInteger object is used, because it allows bit-operations on its value.

- int symbolId
- byte level
- byte symbolType
- byte symbolAttribute
- byte[] clauses = new byte[1]
- BigInteger clausesB
- byte[] flags1 = new byte[1]
- BigInteger flags1B
- int size
- int parentId
- int redefinedId
- short symbolNameLen

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

- String symbolName
- static String symbolAttributeValues[]

This array contains descriptive names for the DataItem.TYPE tag.

Appendix CDocument Type Definition (DTD)

```

1  <?xml version="1.0" encoding="UTF-8" ?>
2
3  <!-- XMI Automatic DTD Generation      -->
4  <!-- Metamodel: DataItem              -->
5
6  <!-- _____ -->
7  <!-- _____ -->
8  <!-- XMI is the top-level XML element for XMI transfer text -->
9  <!-- _____ -->
10
11 <!ELEMENT XMI (XMI.header, XMI.content?, XMI.difference*,
12     XMI.extensions*) >
13 <!ATTLIST XMI
14     xmi.version CDATA #FIXED "1.0"
15     timestamp CDATA #IMPLIED
16     verified (true | false) #IMPLIED
17     >
18
19 <!-- _____ -->
20 <!-- _____ -->
21 <!-- XMI.header contains documentation and identifies the model,
22 <!-- metamodel, and metamodel          -->
23 <!-- _____ -->
24
25 <!ELEMENT XMI.header (XMI.documentation?, XMI.model*, XMI.metamodel*,
26     XMI.metamodelmodel*) >
27
28 <!-- _____ -->
29 <!-- _____ -->
30 <!-- documentation for transfer data      -->
31 <!-- _____ -->
32
33 <!ELEMENT XMI.documentation (#PCDATA | XMI.owner | XMI.contact |
34     XMI.longDescription | XMI.shortDescription |
35     XMI.exporter | XMI.exporterVersion |
36     XMI.notice)* >
37
38 <!ELEMENT XMI.owner ANY >
39
40 <!ELEMENT XMI.contact ANY >
41
42 <!ELEMENT XMI.longDescription ANY >
43
44 <!ELEMENT XMI.shortDescription ANY >

```



```

1  <!ELEMENT XMI.exporter ANY >
2
3  <!ELEMENT XMI.exporterVersion ANY >
4
5  <!ELEMENT XMI.exporterID ANY >
6
7  <!ELEMENT XMI.notice ANY >
8
9  <!-- _____ -->
10 <!-- _____ -->
11 <!-- XMI.element.att defines the attributes that each XML element -->
12 <!-- that corresponds to a metamodel class must have to conform to -->
13 <!-- the XMI specification. -->
14 <!-- _____ -->
15
16 <!ENTITY % XMI.element.att
17     'xmi.id ID #IMPLIED xmi.label CDATA #IMPLIED xmi.uuid
18     CDATA #IMPLIED ' >
19
20 <!-- _____ -->
21 <!-- _____ -->
22 <!-- XMI.link.att defines the attributes that each XML element that -->
23 <!-- corresponds to a metamodel class must have to enable it to -->
24 <!-- function as a simple XLink as well as refer to model -->
25 <!-- constructs within the same XMI file. -->
26 <!-- _____ -->
27
28 <!ENTITY % XMI.link.att
29     'xml:link CDATA #IMPLIED inline (true | false) #IMPLIED
30     actuate (show | user) #IMPLIED href CDATA #IMPLIED role
31     CDATA #IMPLIED title CDATA #IMPLIED show (embed | replace
32     | new) #IMPLIED behavior CDATA #IMPLIED xmi.idref IDREF
33     #IMPLIED xmi.uuidref CDATA #IMPLIED' >
34
35 <!-- _____ -->
36 <!-- _____ -->
37 <!-- XMI.model identifies the model(s) being transferred -->
38 <!-- _____ -->
39
40 <!ELEMENT XMI.model ANY >
41 <!ATTLIST XMI.model
42     %XMI.link.att;
43     xmi.name CDATA #REQUIRED
44     xmi.version CDATA #IMPLIED
45 >
46
47 <!-- _____ -->
48 <!-- _____ -->
49 <!-- XMI.metamodel identifies the metamodel(s) for the transferred -->
50 <!-- data -->

```

```

1  <!-- _____ -->
2  <!-- ELEMENT XMI.metamodel ANY >
   <!-- ATTLIST XMI.metamodel
       %XMI.link.att;
3     xmi.name  CDATA #REQUIRED
       xmi.version CDATA #IMPLIED
4   >
5  <!-- _____ -->
6  <!-- _____ -->
   <!-- XMI.metamodel identifies the metamodel(s) for the
   <!-- transferred data
   <!-- _____ -->
7  <!-- ELEMENT XMI.metamodel ANY >
   <!-- ATTLIST XMI.metamodel
       %XMI.link.att;
8     xmi.name  CDATA #REQUIRED
       xmi.version CDATA #IMPLIED
9   >
10 <!-- _____ -->
11 <!-- _____ -->
   <!-- XMI.content is the actual data being transferred
   <!-- _____ -->
12 <!-- ELEMENT XMI.content ANY >
13 <!-- _____ -->
14 <!-- _____ -->
   <!-- XMI.extensions contains data to transfer that does not conform
   <!-- to the metamodel(s) in the header
   <!-- _____ -->
15 <!-- ELEMENT XMI.extensions ANY >
16 <!-- ATTLIST XMI.extensions
       xmi.extender CDATA #REQUIRED
17 >
18 <!-- _____ -->
19 <!-- _____ -->
   <!-- extension contains information related to a specific model
   <!-- construct that is not defined in the metamodel(s) in the header
   <!-- _____ -->
20 <!-- ELEMENT XMI.extension ANY >
   <!-- ATTLIST XMI.extension
       %XMI.element.att;
21      %XMI.link.att;

```

```

1      xmi.extender CDATA #REQUIRED
      xmi.extenderID CDATA #REQUIRED
2      >
3      <!-- _____ -->
      <!-- _____ -->
      <!-- XMI.difference holds XML elements representing differences to a -->
4      <!-- base model -->
      <!-- _____ -->
5      <!ELEMENT XMI.difference (XMI.difference | XMI.delete | XMI.add |
      XMI.replace)* >
6      <!ATTLIST XMI.difference
      %XMI.element.att;
7      %XMI.link.att;
      >
8      <!-- _____ -->
      <!-- _____ -->
9      <!-- XMI.delete represents a deletion from a base model -->
      <!-- _____ -->
10     <!ELEMENT XMI.delete EMPTY >
      <!ATTLIST XMI.delete
11         %XMI.element.att;
         %XMI.link.att;
12     >
13     <!-- _____ -->
      <!-- _____ -->
      <!-- XMI.add represents an addition to a base model -->
14     <!-- _____ -->
15     <!ELEMENT XMI.add ANY >
      <!ATTLIST XMI.add
16         %XMI.element.att;
         %XMI.link.att;
         xmi.position CDATA "-1"
17     >
18     <!-- _____ -->
      <!-- _____ -->
      <!-- XMI.replace represents the replacement of a model construct -->
19     <!-- with another model construct in a base model -->
      <!-- _____ -->
20     <!ELEMENT XMI.replace ANY >
      <!ATTLIST XMI.replace
21         %XMI.element.att;
         %XMI.link.att;

```

```

1      xmi.position CDATA "-1"
2      >
3      <!-- _____ -->
4      <!-- _____ -->
5      <!-- XMI.reference may be used to refer to data types not defined in -->
6      <!-- the metamodel -->
7      <!-- _____ -->
8      <!-- _____ -->
9      <!-- _____ -->
10     <!-- _____ -->
11     <!-- This section contains the declaration of XML elements -->
12     <!-- representing data types -->
13     <!-- _____ -->
14     <!-- _____ -->
15     <!-- _____ -->
16     <!-- _____ -->
17     <!-- _____ -->
18     <!-- _____ -->
19     <!-- _____ -->
20     <!-- _____ -->
21     <!-- _____ -->

```

```

1          XMI.CorbaTcChar | XMI.CorbaTcWchar |
          XMI.CorbaTcOctet | XMI.CorbaTcAny |
2          XMI.CorbaTcTypeCode | XMI.CorbaTcPrincipal |
          XMI.CorbaTcNull | XMI.CorbaTcVoid |
          XMI.CorbaTcLongLong |
3          XMI.CorbaTcLongDouble) >
<!ATTLIST XMI.CorbaTypeCode
4          %XMI.element.att;
>

5 <!ELEMENT XMI.CorbaTcAlias (XMI.CorbaTypeCode) >
<!ATTLIST XMI.CorbaTcAlias
6          xmi.tcName CDATA #REQUIRED
          xmi.tcId CDATA #IMPLIED
7 >

<!ELEMENT XMI.CorbaTcStruct (XMI.CorbaTcField)* >
8 <!ATTLIST XMI.CorbaTcStruct
          xmi.tcName CDATA #REQUIRED
          xmi.tcId CDATA #IMPLIED
9 >

10 <!ELEMENT XMI.CorbaTcField (XMI.CorbaTypeCode) >
<!ATTLIST XMI.CorbaTcField
11          xmi.tcName CDATA #REQUIRED
>

12 <!ELEMENT XMI.CorbaTcSequence (XMI.CorbaTypeCode |
          XMI.CorbaRecursiveType) >
13 <!ATTLIST XMI.CorbaTcSequence
          xmi.tcLength CDATA #REQUIRED
14 >

<!ELEMENT XMI.CorbaRecursiveType EMPTY >
15 <!ATTLIST XMI.CorbaRecursiveType
          xmi.offset CDATA #REQUIRED
16 >

<!ELEMENT XMI.CorbaTcArray (XMI.CorbaTypeCode) >
17 <!ATTLIST XMI.CorbaTcArray
          xmi.tcLength CDATA #REQUIRED
18 >

<!ELEMENT XMI.CorbaTcObjRef EMPTY >
19 <!ATTLIST XMI.CorbaTcObjRef
          xmi.tcName CDATA #REQUIRED
          xmi.tcId CDATA #IMPLIED
20 >

21 <!ELEMENT XMI.CorbaTcEnum (XMI.CorbaTcEnumLabel) >

```

```
1  <![ATTLIST XMI.CorbaTcEnum
   xmi.tcName CDATA #REQUIRED
   xmi.tcId CDATA #IMPLIED
2  >
3  <![ELEMENT XMI.CorbaTcEnumLabel EMPTY >
   <![ATTLIST XMI.CorbaTcEnumLabel
4     xmi.tcName CDATA #REQUIRED
   >
5  <![ELEMENT XMI.CorbaTcUnionMbr (XMI.CorbaTypeCode, XMI.any) >
   <![ATTLIST XMI.CorbaTcUnionMbr
6     xmi.tcName CDATA #REQUIRED
   >
7  <![ELEMENT XMI.CorbaTcUnion (XMI.CorbaTypeCode, XMI.CorbaTcUnionMbr*) >
   <![ATTLIST XMI.CorbaTcUnion
8     xmi.tcName CDATA #REQUIRED
     xmi.tcId CDATA #IMPLIED
9  >
10 <![ELEMENT XMI.CorbaTcExcept (XMI.CorbaTcField)* >
   <![ATTLIST XMI.CorbaTcExcept
11     xmi.tcName CDATA #REQUIRED
     xmi.tcId CDATA #IMPLIED
   >
12 <![ELEMENT XMI.CorbaTcString EMPTY >
   <![ATTLIST XMI.CorbaTcString
13     xmi.tcLength CDATA #REQUIRED
   >
14 <![ELEMENT XMI.CorbaTcWstring EMPTY >
   <![ATTLIST XMI.CorbaTcWstring
15     xmi.tcLength CDATA #REQUIRED
   >
16 <![ELEMENT XMI.CorbaTcFixed EMPTY >
   <![ATTLIST XMI.CorbaTcFixed
17     xmi.tcDigits CDATA #REQUIRED
     xmi.tcScale CDATA #REQUIRED
18 >
19 <![ELEMENT XMI.CorbaTcShort EMPTY >
20 <![ELEMENT XMI.CorbaTcLong EMPTY >
   <![ELEMENT XMI.CorbaTcUshort EMPTY >
21 <![ELEMENT XMI.CorbaTcUlong EMPTY >
```

1 <!ELEMENT XMI.CorbaTcFloat EMPTY >
 2 <!ELEMENT XMI.CorbaTcDouble EMPTY >
 3 <!ELEMENT XMI.CorbaTcBoolean EMPTY >
 4 <!ELEMENT XMI.CorbaTcChar EMPTY >
 5 <!ELEMENT XMI.CorbaTcWchar EMPTY >
 6 <!ELEMENT XMI.CorbaTcOctet EMPTY >
 7 <!ELEMENT XMI.CorbaTcAny EMPTY >
 8 <!ELEMENT XMI.CorbaTcTypeCode EMPTY >
 9 <!ELEMENT XMI.CorbaTcPrincipal EMPTY >
 10 <!ELEMENT XMI.CorbaTcNull EMPTY >
 11 <!ELEMENT XMI.CorbaTcVoid EMPTY >
 12 <!ELEMENT XMI.CorbaTcLongLong EMPTY >
 13 <!ELEMENT XMI.CorbaTcLongDouble EMPTY >
 14 <!-- _____ -->
 15 <!-- _____ -->
 16 <!-- METAMODEL: DataItem _____ -->
 17 <!-- _____ -->
 18 <!ELEMENT DataItem.child (DataItem)* >
 19 <!-- _____ -->
 20 <!-- _____ -->
 21 <!-- METAMODEL CLASS: DataItem _____ -->
 22 <!-- _____ -->
 23 <!-- _____ -->
 24 <!-- METAMODEL CLASS: DataItem _____ -->
 25 <!-- _____ -->
 26 <!-- _____ -->
 27 <!-- METAMODEL CLASS: DataItem _____ -->
 28 <!-- _____ -->
 29 <!-- _____ -->
 30 <!-- METAMODEL CLASS: DataItem _____ -->
 31 <!-- _____ -->
 32 <!-- _____ -->
 33 <!-- METAMODEL CLASS: DataItem _____ -->
 34 <!-- _____ -->
 35 <!-- _____ -->
 36 <!-- METAMODEL CLASS: DataItem _____ -->
 37 <!-- _____ -->
 38 <!-- _____ -->
 39 <!-- METAMODEL CLASS: DataItem _____ -->
 40 <!-- _____ -->
 41 <!-- _____ -->
 42 <!-- METAMODEL CLASS: DataItem _____ -->
 43 <!-- _____ -->
 44 <!-- _____ -->
 45 <!-- METAMODEL CLASS: DataItem _____ -->
 46 <!-- _____ -->
 47 <!-- _____ -->
 48 <!-- METAMODEL CLASS: DataItem _____ -->
 49 <!-- _____ -->
 50 <!-- _____ -->
 51 <!-- METAMODEL CLASS: DataItem _____ -->
 52 <!-- _____ -->
 53 <!-- _____ -->
 54 <!-- METAMODEL CLASS: DataItem _____ -->
 55 <!-- _____ -->
 56 <!-- _____ -->
 57 <!-- METAMODEL CLASS: DataItem _____ -->
 58 <!-- _____ -->
 59 <!-- _____ -->
 60 <!-- METAMODEL CLASS: DataItem _____ -->
 61 <!-- _____ -->
 62 <!-- _____ -->
 63 <!-- METAMODEL CLASS: DataItem _____ -->
 64 <!-- _____ -->
 65 <!-- _____ -->
 66 <!-- METAMODEL CLASS: DataItem _____ -->
 67 <!-- _____ -->
 68 <!-- _____ -->
 69 <!-- METAMODEL CLASS: DataItem _____ -->
 70 <!-- _____ -->
 71 <!-- _____ -->
 72 <!-- METAMODEL CLASS: DataItem _____ -->
 73 <!-- _____ -->
 74 <!-- _____ -->
 75 <!-- METAMODEL CLASS: DataItem _____ -->
 76 <!-- _____ -->
 77 <!-- _____ -->
 78 <!-- METAMODEL CLASS: DataItem _____ -->
 79 <!-- _____ -->
 80 <!-- _____ -->
 81 <!-- METAMODEL CLASS: DataItem _____ -->
 82 <!-- _____ -->
 83 <!-- _____ -->
 84 <!-- METAMODEL CLASS: DataItem _____ -->
 85 <!-- _____ -->
 86 <!-- _____ -->
 87 <!-- METAMODEL CLASS: DataItem _____ -->
 88 <!-- _____ -->
 89 <!-- _____ -->
 90 <!-- METAMODEL CLASS: DataItem _____ -->
 91 <!-- _____ -->
 92 <!-- _____ -->
 93 <!-- METAMODEL CLASS: DataItem _____ -->
 94 <!-- _____ -->
 95 <!-- _____ -->
 96 <!-- METAMODEL CLASS: DataItem _____ -->
 97 <!-- _____ -->
 98 <!-- _____ -->
 99 <!-- METAMODEL CLASS: DataItem _____ -->
 100 <!-- _____ -->

1 <!ATTLIST DataItem.hasValue
xmi.value (true | false) #REQUIRED
2 >
3 <!ELEMENT DataItem.Value (#PCDATA | XMI.reference)* >
4 <!ELEMENT DataItem.parent (DataItem)? >
5 <!ELEMENT DataItem (DataItem.Name?, DataItem.Type?, DataItem.Length?,
DataItem.hasValue?, DataItem.Value?, XMI.extension*,
6 DataItem.parent?, DataItem.child*)? >
7 <!ATTLIST DataItem
%XMI.element.att;
%XMI.link.att;
8 >
9
10
11
12
13
14
15
16
17
18
19
20
21